

APPEAL BRIEF

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

**In re Application of: PHENIX, John**

**Application No.: 10/667,816**

**Filed: September 22, 2003**

**Title: Utility for Identifying Differences Between  
Java Objects**

**Examiner: RADTKE, Mark**

**TC/Art Unit: 2165**

**Confirmation No.: 2173**

---

Mail Stop **APPEAL BRIEF - PATENTS**  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**APPELLANT'S BRIEF PURSUANT TO 37 C.F.R. § 41.37**

Sir:

Appellant submits this brief in accordance with 37 C.F.R. § 41.37.

**I. REAL PARTY IN INTEREST**

The real party in interest is JP MORGAN CHASE BANK NA (Assignee) by virtue of an assignment executed by the inventor (Appellant) to JP MORGAN CHASE BANK NA (recorded by the Assignment Branch of the U.S. Patent and Trademark Office on September 22, 2003 at Reel/Frame 014547/0862).

**II. RELATED APPEALS AND INTERFERENCES**

None.

### **III. STATUS OF CLAIMS**

In the application under appeal, Claims 1, 3-5, and 7-9 stand rejected based on 35 U.S.C. § 101 for allegedly being directed to non-statutory subject matter. Claims 1, 3-5, and 7-9 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,826,716 (*Mason*) in view of "Java Performance Tuning," Section 7.5, "Recursion and Stacks," September 2000 (*Shirazi*) and further in view of U.S. Patent No. 6,662,312 (*Keller et al.*). The rejections of claims 1, 3-5, and 7-9 are appealed.

### **IV. STATUS OF AMENDMENTS**

Appellant has submitted no amendments after the final rejection. All amendments prior to the close of prosecution on the merits have been entered.

### **V. SUMMARY OF CLAIMED SUBJECT MATTER**

Independent claim 1 of the present application is directed to a computer implemented method for "comparing a first object and a second object in an object-oriented operating system." The method can include, *inter alia*, (a) "determining whether the first object is equal to the second object; and if said objects are not equal," the method includes the steps of (b) "obtaining one or more methods from said first object and said second object," (c) "determining whether the one or more methods from said first object are equal to the one or more methods from said second object." In step (d), the method "recursively performs steps (b) and (c) until all of the methods for the first object and the second object have been obtained."

The method also includes (e) "generating a document comprising a listing of differences between the methods," (f) "transforming the document into a human-readable form," and (g) "displaying the human-readable form to a user."

Thus, as described in the specification of the present application, to solve the problem of there presently being no universally defined method to determine the differences between two objects, the claimed method beneficially provides for a utility for comparing two objects in an object-oriented operating system and recording the differences so that they may be put in human-readable form.

"Determining whether the first object is equal to the second object," is described in the specification at least at FIG. 1 reference 154 "a.compareTo(b)" and reference 156 "a.equals(b)" and in similar code steps in FIG. 7, and on page 2, paragraph [0019] of U.S. Patent Application Publication No. 2004/0221264: "two Java objects are compared by calling one of the Java equality methods, "Comparable.compareTo( )" or "Objects.equals( )," which are known in the art. While the exemplary embodiment of this invention is described using Java, one skilled in the art will appreciate that this invention can be implemented in any object-oriented language that supports method introspection after studying this specification. According to the exemplary embodiment of this invention, the Java equality method "Comparable.compareTo( )" is called if the objects to compare are instances of Comparable. Otherwise, the "Objects.equals( )" method is called." The step of "if said objects are not equal, obtaining one or more methods from said first object and said second object, determining whether the one or more methods from said first object are equal to the one or more methods from said second object" is supported at least at page 2, paragraph [0019]: "If the selected equality method indicates that there is a difference between the two objects, then get . . . ( ) methods of each object are invoked in turn." The step of "determining whether the one or more methods from said first object are equal to the one or more methods from said second object" is supported at least at page 2, paragraph [0019]: "results of the get . . . ( ) methods are compared." The step of "recursively performs steps (b) and (c) until all of the methods for the first object and the second object have been obtained" is

supported at least at page 2, paragraph [0019]: “Importantly, the get . . . ( ) method is recursively invoked until the Class of the result has no more get . . . ( ) methods to decompose.” The step of “generating a document comprising a listing of differences between the methods,” is supported at least at FIG. 5 and page 2, paragraph [0019]: “If there are differences between the methods, the differences are stored in an XML document.” The step of “transforming the document into a human-readable form,” is supported at least at page 1, paragraph 8: “Advantageously, the objects to be compared are Java objects and the differences are recorded in XML format, which is easily transformed into human-understandable form (e.g., transform the XML into web pages using XSL).” The step of “displaying the human-readable form to a user” is supported at least at page 1, paragraph 8 as just described and at page 3, paragraph [0033]: “This XML document may be used by, for example, a browser, which displays the differences in human-readable form.”

References in this brief to supporting portions of the specification and drawings are given to provide exemplary embodiments, not to provide limitations to the claims.

## **VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL**

Appellant respectfully requests the Board of Patent Appeals and Interferences review the following grounds of rejection on appeal:

1. Whether claims 1, 3-5, and 7-9 are statutory subject matter under 35 U.S.C. § 101.
2. Whether claims 1, 3-5, and 7-9 are patentable under 35 U.S.C. 103(a) over *Mason* in view of *Shirazi* and further in view of *Keller et al.*

## **VII. ARGUMENT**

Appellant respectfully submits that claims 1, 3-5, and 7-9 are directed to statutory subject matter, are in proper form, and are patentable over the prior art of record.

### **Claims 1, 3-5, and 7-9 Are Statutory Subject Matter Under 35 U.S.C. § 101**

Claims 1, 3-5, and 7-9 stand rejected based on 35 U.S.C. § 101 for allegedly being directed to non-statutory subject matter. Appellant respectfully disagrees.

The first part of the rejection contends that the claimed method does not produce a useful, concrete and tangible result. More specifically, the Examiner asserts that the claimed subject matter fails to produce an assured, repeatable result because certain inputs would not produce any output. More specifically, if the method in step (a) of independent claim 1 determines that two objects are equal, then the method terminates without a result. Also, if the objects are not determined to be equal in step (a), and step (e) of independent claim 1 shows no differences between the objects, then again the program would terminate without a result.

Appellant respectfully disagrees with the Examiner's position. Many operating system commands and other commands written for execution on a UNIX-type platform deliberately produce no visible results specifically when the command succeeds, and this lack of output is an indication to the user of a particular state of affairs. For example, the UNIX diff command produces no output specifically when it finds no differences between two inputted documents to be compared and produces an output containing differences between the documents when the inputted documents are not the same. Likewise, when the claimed invention produces a null output, this is an indication that the documents are "equal" according to the criteria of claim 1, which would be familiar to a user who has experience with UNIX, and text output is produced when there are differences, similar to the UNIX diff command.

Therefore, Appellant asserts that producing a null output is a useful, concrete and tangible result, indicating that two documents are equal, and therefore independent claim 1 is directed to statutory subject matter under 35 U.S.C. § 101.

Thus, Appellant asserts that claim 1 and claims 3-5, and 7-9 dependent therefrom are directed to statutory subject matter. Thus, for the reasons stated above, Appellant believes that the Examiner's rejection of claims 1, 3-5, and 7-9 under 35 U.S.C. § 101 should be reversed and such action is respectfully requested.

**Claims 1, 3-5, and 7-9 are patentable over Mason in view of Shirazi and further in view of Keller et al.**

Claims 1, 3-5, and 7-9 was rejected under 35 U.S.C. 103(a) as being unpatentable over *Mason* in view of *Shirazi* and further in view of *Keller et al.* Appellant respectfully disagrees.

*Mason* describes a system for generating "test programs" for testing Java Web applications. (*Mason* at Abstract). The generated test programs are designed to test "more combinations of actions, web application configurations or scenarios." (*Id.* at col. 2, lns. 55-57). *Mason* describes identifying modules, and identifying a "QOS [Quality of Service] element." (*Id.* at col. 2, lns. 66-67 to col. 3, lns. 1-2). Also described, are identifying a "user identification and a user password ... and testing the software resource to be tested by use of the Java test code, including passing as parameters to the Java test code at run time the user identification and user password." (*Id.* at col. 3, lns. 3-9).

*Mason* further describes extracting data from a deployment descriptor to facilitate generation of the Java test code. (*Id.* at col. 12, lns. 23-25). "In typical embodiments, testing the software resource further includes creating ...an instance of the JavaBean (450), invoking (448), in the created instance, (450), the software resource to be tested ... and reporting (444)

whether invoking the protected JavaBean method succeeded." (*Id.* at col. 12, lns. 40-46 and FIG. 4)(emphasis added). The test code described by *Mason* can generate different kinds of tests, "each targeted at a specific aspect of quality or verification. For instance, correctness, load/stress test, reliability, long running tests, and so on." (*Id.* at col. 15, lns. 32-35). Thus, *Mason* describes a system for generating a test program for testing a Java Web application to determine if it succeeds or fails.

Appellant asserts that *Mason* does not teach, suggest or provide motivation for all of the features recited by amended claim 1 of the present application. While claim 1 of the present application claims a method for "comparing a first object and a second object in an object-oriented operating system" and "generating a document comprising a listing of differences between the methods," in stark contrast, *Mason* describes a system for testing various parameters of Java Web applications to determine if the applications succeed. Thus, the system described by *Mason* is very different from the method claimed by claim 1 of the present application.

Moreover, Appellant asserts that *Mason* does not describe other features recited by claim 1. For example, *Mason* does not teach, suggest or provide motivation for "determining whether the first object is equal to the second object; and if said objects are not equal," or "obtaining one or more methods from said first object and said second object." In addition, *Mason* does not teach, suggest or provide motivation for the recited "determining whether the one or more methods from said first object are equal to the one or more methods from said second object," "generating a document comprising a listing of differences between the methods," "transforming the document into a human-readable form," or "displaying the human-readable form to a user."

The Examiner cited FIG. 4, element 412, FIG. 4, element 406 and column 12, lines 20-23 of *Mason* as teaching "Determining whether the first object is equal to the second

object.” FIG. 4, element 412 refers to an XML tag “<enterprise-bean>” and element 406 refers to XML tag “<ejb-name>.” There is no teaching or suggestion in the specification of *Mason* when referring to these tags as comparing them in any way for equality. Column 12, lines 20-23 says there is no need in *Mason* for objects to support introspection and a deployment descriptor itself provides all the information needed to generate test code. Thus, this section of *Mason* does not refer to determining equality of objects but refers only to generating code.

The Examiner cited column 12, lines 40-46 of *Mason* as teaching “obtaining one or more methods from said first object and said second object.” This section of *Mason* refers to testing software resources by creating a JavaBean, invoking the resource (Java bean method) to be tested, and reporting whether invoking a protected Java Bean method succeeded or not. This has no connection whatsoever with obtaining methods to test for equality to be compared between a first object and a second object. Here, *Mason* is referring only to one object, i.e. the Java Bean that was created.

The Examiner cited column 12, line 45 “reporting (444) whether the protected JavaBean method succeeded” as teaching “determining whether the one or more methods from said first object are equal to the one or more methods from said second object.” As already stated above, this portion of *Mason*, in context, refers only to testing a software resource creating a Java Bean object and invoking a protected Java Bean method, then **reporting** whether invoking the protected JavaBean method succeeded or not. This has nothing to do with testing the equality of methods of two objects to be tested.

The Examiner cited column 8, line 47 of *Mason* as teaching “recursively performs steps (b) and (c) until all of the methods for the first object and the second object have been obtained.” Column 8, line 47 of *Mason* refers to sample code for testing an enterprise java bean (EJB). Once EJB code has been parsed, for each method permission element, open a Java .class

file for the EJB and reflect on the EJB class to discover a method signature and return type. Then generate Java code to test a protected EJB method. This section of *Mason* has to do with looking at source code of a single Java Bean and generating test code to test its methods, not recursively testing for equality between methods of two objects. There is only one object involved in *Mason*.

The Examiner cited column 12, line 45 and column 15, lines 18-20 as teaching "generating a document comprising a listing of differences between the methods." As discussed above, column 12, line 45 refers to a single Java Bean object whose protected method is invoked and tested for success. Column 15, lines 18-20 refer to printing a stack trace if an exception is thrown in the code which tests a protected EJB method. There is no description of generating a document that lists the difference between two methods of two objects. As stated above, *Mason* teaches one object and is not making a comparison. Printing a stack trace is not printing difference between two objects.

Thus, Appellant asserts that *Mason* fails to teach or disclose several elements or steps of Claim 1.

*Shirazi* describes the use of programming techniques to convert recursive method calls to iterative method calls. (*Shirazi* at page 1). *Shirazi* does not make up for the shortcomings of *Mason*. For example, *Shirazi*, either alone, or in combination with *Mason*, does not teach, suggest, or provide motivation for "comparing a first object and a second object in an object-oriented operating system." Nor does any *Mason-Shirazi* combination teach, suggest or provide motivation for "determining whether the first object is equal to the second object; and if said objects are not equal," "obtaining one or more methods from said first object and said second object," and "determining whether the one or more methods from said first object are equal to the one or more methods from said second object." Still further, the *Mason-Shirazi* combination

does not teach, suggest or provide motivation for "generating a document comprising a listing of differences between the methods," "transforming the document into a human-readable form." and "displaying the human-readable form to a user."

Accordingly, for at least these reasons, Appellant deems claim 1 to distinguish patentably over any hypothetical *Mason-Shirazi* combination.

Appellant asserts that *Keller* fails to correct the deficiencies of either *Mason* or *Shirazi*. *Keller* is directed to a software testing automation system for testing a plurality of deployed images that are spread across multiple software platforms. Each deployed image includes a test component configured to accept a connection on a known testing port, and the test component for a deployed image is inserted in a development environment and is then integrated into the image upon deployment. The system includes a test engine configured to run a plurality of tests on the plurality of deployed images. The test engine runs a test on an image under test by requesting a connection to the corresponding test component on the known testing port and sending commands over the connection. A user interface connected to the test engine allows a user to control the test engine and to view test results.

Appellant asserts that *Keller* fails to disclose: "comparing a first object and a second object in an object-oriented operating system," "determining whether the first object is equal to the second object; and if said objects are not equal," (b) "obtaining one or more methods from said first object and said second object," (c) "determining whether the one or more methods from said first object are equal to the one or more methods from said second object." Accordingly, for at least these reasons, claim 1 is deemed to distinguish patentably over any hypothetical *Mason-Shirazi-Keller* combination. Claims 3-5, and 7-9 depend from, and further narrow and define, claim 1, that has been discussed above and is believed to be allowable over any *Mason-Shirazi-Keller* combination. Accordingly, for at least these reasons, claims 3-5, and

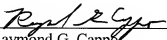
7-9 are deemed to distinguish patentably over any hypothetical *Mason-Shirazi-Keller* combination.

Thus, appellant submits that claim 1 and claims 3-5, and 7-9 dependent therefrom are patentable over each of the references of record, either taken alone, or in any proposed hypothetical combination. For the reasons stated above, Appellant believes that the Examiner's rejection of claims 1, 3-5, and 7-9 under 35 U.S.C. § 103(a) should be reversed and such action is respectfully requested.

**CONCLUSION**

For the reasons stated above, claims 1, 3-5, and 7-9 are directed to statutory subject matter, are patentable over the prior art of record, and the rejections of these claims under 35 U.S.C. § 101 and 35 U.S.C. § 103(a) are improper and should be withdrawn. Appellant respectfully asks the board to reverse the Examiner's rejections with instructions to allow the claims.

If any fees are deemed necessary for this Appeal Brief to be entered and considered, then the Commissioner is authorized to charge such fee to Deposit Account No. 501358. Appellant's undersigned agent may be reached by telephone at (973) 597-2500. All correspondence should continue to be directed to our address listed below.

  
\_\_\_\_\_  
Raymond G. Capp6  
Registration No. 53,836  
Registered Patent Agent for  
Applicants

DOCKET ADMINISTRATOR  
LOWENSTEIN SANDLER PC  
65 Livingston Avenue  
Roseland, NJ 07068

## **VIII. CLAIMS APPENDIX**

1. (previously presented) A computer implemented method for comparing a first object and a second object in an object-oriented operating system comprising the steps of:

(a) determining whether the first object is equal to the second object; and

if said objects are not equal:

(b) obtaining one or more methods from said first object and said second object;

(c) determining whether the one or more methods from said first object are equal to the one or more methods from said second object; and

(d) recursively performing steps (b) and (c) until all of the methods for the first object and the second object have been obtained;

(e) generating a document comprising a listing of differences between the methods;

(f) transforming the document into a human-readable form; and

(g) displaying the human-readable form to a user.

2. (canceled)

3. (previously presented) A method in accordance with claim 1 wherein step (b) further comprises storing names of said methods, step (c) comprises storing indicia representing the determination whether the methods are the same and step (e) comprises generating a document from said stored names and indicia.

4. (previously presented) A method in accordance with claim 1 wherein said object-oriented operating system comprises JAVA and step (a) comprises running an equality method on said objects.

5. (previously presented) A method in accordance with claim 1 wherein said object-oriented operating system comprises JAVA and step (c) comprises running an equality method on said objects.

6. (canceled)

7. (previously presented) A method in accordance with claim 1 wherein human-readable form comprises XML.

8. (previously presented) A method in accordance with claim 1 wherein transforming the document into human-readable form comprises transforming the document into a web page.

9. (previously presented) A method in accordance with claim 1 wherein said object-oriented operating system comprises JAVA and step (b) comprises invoking a get...() method of each object.

10. (canceled)

**IX. EVIDENCE APPENDIX**

U.S. Patent No. 6,826,716 (*Mason*) cited by the Examiner in Final Office Action mailed on March 8, 2007, and in Office Action mailed on September 8, 2006.

"Java Performance Tuning," Section 7.5, "Recursion and Stacks," September 2000 (*Shirazi*) cited by the Examiner in Final Office Action mailed on March 8, 2007, and in Office Action mailed on September 8, 2006.

U.S. Patent No. 6,662,312 (*Keller et al.*) cited by the Examiner in Final Office Action mailed on March 8, 2007, and in Office Action mailed on September 8, 2006.

**X. RELATED PROCEEDINGS APPENDIX**

None.